OCCi

OPEN CLOUD COMPUTING INTERFACE

Sam Johnston <samj@samj.net> http://samj.net/
Secretary - OCCI-WG
Open Grid Forum (OGF)

Why are we here?

- To establish a thriving marketplace and protect user freedoms
- To mitigate risks such as those highlighted by the Danger/Sidekick debacle
- To make Open Cloud a reality:

open cloud™

What is Open Cloud?

- Open Cloud Initiative (OCI) launched in March 2009, modelled after OSI/OSD
- Open Cloud Principles (OCP) which cover four main points:
    - **Open Formats**
    - **Open Interfaces**
    - Open Data
    - Open Source

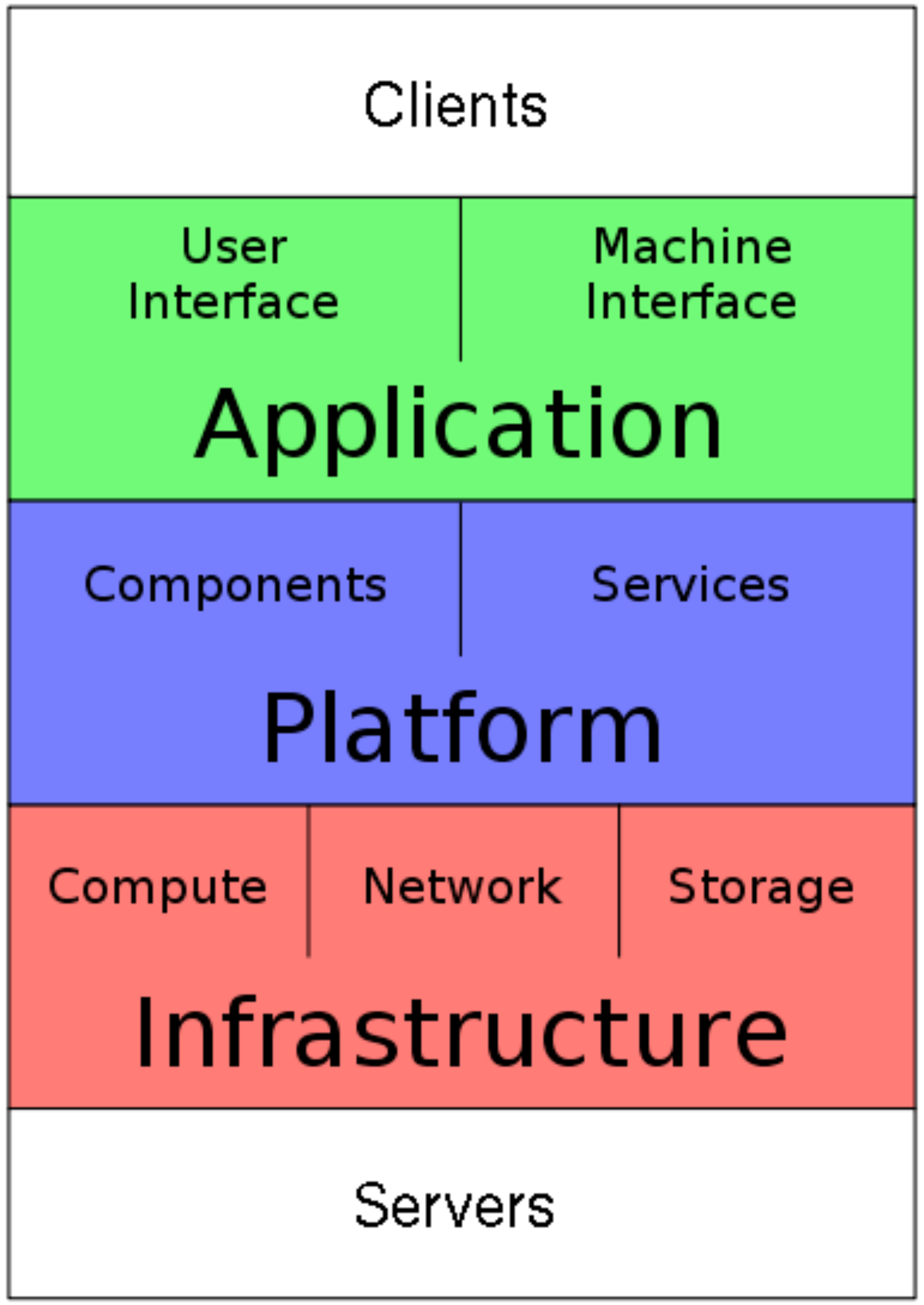
Open Cloud

- Two key requirements:
  - “Open Formats: All pertinent functionality must be exposed by way of Open Standard Interfaces.”
  - “Open Interfaces: All user data and metadata must be represented in Open Standard formats.”
- No point having unfettered access to opaque data
- No point having no access to transparent data
- Targets all cloud computing products - vendors can keep their secret sauce
- Guarantees critical freedoms without stifling innovation/ability to differentiate

Open Standards

- **Copyrights:** The standard must be documented in all its details, published and both accessible and [re]usable free of charge.
- **Patents:** Any patents possibly present on [parts of] the standard must be irrevocably made available on a royalty-free basis.
- **Trademarks:** Any trademarks possibly present on identifier(s) must be used for non-discriminatory enforcement of compliance only.
- **Implementations:** There must be multiple full, faithful and interoperable implementations (for both client and server where applicable) and at least one such implementation must be licensed under an Open Source Initiative (OSI) Approved license or placed into the public domain.

Cloud Computing Stack



Cloud Computing Stack

Strategy

- Start at the bottom & work our way up (with a common core protocol)
    - Infrastructure - focus of cloud discussions today (fixed scope):
        - Compute, Network, Storage
    - Platforms - focus of cloud discussions tomorrow (broader scope):
        - Databases, Queues, Runtimes, Others (MTurk)
    - Applications - future of cloud computing (unlimited scope):

Proposed Roadmap

- OGF 27 (October 12-15, 2009)
  - Delivery of OCCI Core & OCCI Infrastructure drafts
- OGF 28 (March 15-19, 2010)
  - Delivery of OCCI Core & OCCI Infrastructure finals, OCCI Platform draft
- OGF 29 (October 2010)
  - Delivery of OCCI Platform final & OCCI Application draft (final early 2011)

Traditional Standards Process

- Define an abstract model & map it to one or more concrete formats
  - Quick & dirty way to get standards out the door quickly, but...
  - Rigid & brittle standards result
  - Changes require “rinse & repeat” - breaks existing clients
  - Can be very difficult to extend - especially for third parties
- Better: use existing standards like AtomPub (like Google did with GData)
- Or develop a clean core (ala HTTP) that can be easily & safely extended

What’s so good about OCCI?

- It’s RESTful, and not in an RPC-hybrid kind of way.
- It’s clean, like really clean. We deviate from HTTP only where absolutely necessary so existing features such as caching & partial GETs just work.
  - Envelope based formats such as Atom & SOAP interfere with the payload which requires extra complexity & processing to unwrap.
- It’s a standard that’s standards-based. We use existing IETF standards in strong preference to writing our own (in which case we use Internet-Drafts)
  - We’re also format agnostic, just like Internet today (e.g. GIF vs JPG vs PNG vs SVG). Let the experts (e.g. DMTF) define their formats (e.g. OVF).

Future Extensibility

- It’s early days for cloud standards:
    - Sufficient standardisation for interoperability & portability needs today
    - Extreme extensibility to cater for future needs e.g. teleporting workloads ;)
- Anyone can extend the interface by writing a short specification document
- Anyone can declare new actions, attributes & kinds in their own namespace
- We can all benefit from by way of community maintained registries ala IANA
- Compatible with focused SDOs like SNIA (Storage), DMTF (Virtualisation), etc.

Extensibility Examples

- Cloud infrastructure service requires advanced networking (MPLS labels):
  - Developer sends attribute request to mailing list, added to registry.
- Cloud platform service requires support for database checkpointing:
  - Developer sends description of action to mailing list, added to registry.
- Cloud application service requires modelling of payroll tasks:
  - Developer writes specification and lists OCCI as normative reference.

How does it work?

- You wouldn’t believe how easy it is for both **machines** and **users**...
  - GET / HTTP/1.1 -> Collection of resources in text/uri-list format (1 per line)
  - GET /vapp/web01 HTTP/1.1 -> Resource in native format (eg OVF for VM)
    - Related resources, actions, categories & metadata in HTTP headers
    - PUT resource back in native format to alter it
  - POST /vapp/web01;start HTTP/1.1 -> Actions triggered by HTTP POST
    - Standard web forms used for parametrised actions (e.g. “resize”)

Interfaces

- User Interface (application/xhtml+xml)
  - Resources in XHTML5 format (human friendly, ok for computers too!)
  - Lowers learning curve and enables semantic web compatibility
- Machine Interface (all other content types)
  - Resources in native format like OVF (machine, not human friendly)
  - Trying to keep HTTP body “8-bit clean”; no envelopes or alternative types
  - Existing user agents (LWP, httplib, etc.) work out of the box!

Web Linking (draft-nottingham-http-link-header)

- Web linking was originally specified in HTTP (now being revitalised)
- LINKs defined by Atom, HTML & HTTP - XML-based example below:
  - <LINK REL=”related” HREF=”/docs.pdf” TYPE=”application/pdf” />
- Extensible so we can add attributes on link relationships (many-many join)
  - Use this for network & storage links (e.g. for device name ‘eth0’, address)
- Web links can point at anything: related resources, alternative representations, first/previous/next/last, supporting APIs (e.g. CDMI)

Web Categories (draft-johnston-http-category-header)

- Tackles organisation of web resources into categories (modelled after Atom)
- Supports unlimited vocabularies (“schemes”), e.g. animal/vegetable/mineral
- Used in OCCI Core to capture resource “kinds” (compute/storage/network)
- OCCI will define schemes for interoperability (e.g. OS, CPU arch, patch level)
- End users can define their own schemes (e.g. us-east, eu-west)
- Query interface (courtesy GData) allows us to ask questions like “Show me all the Windows machines on the US East cost that are missing Q123456 patch” (e.g. “/-/windows/us-east/-q123456”)

Example Transaction

```
> GET /us-east/webapp/vm01 HTTP/1.1
> User-Agent: occi-client/1.0 (linux) libcurl/7.19.4 OCCI/1.0
> Host: cloud.example.com
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Sat, 10 Oct 2009 12:56:51 GMT
< Content-Type: application/ovf
< Link: </us-east/webapp/vm01;start>;
<       rel="http://purl.org/occi/action/start";
<       title="Start"
< Link: </us-east/webapp/build.pdf>;
<       rel="related";
<       title="Documentation";
<       type="application/pdf"
< Category: compute;
<       label="Compute Resource";
<       scheme="http://purl.org/occi/kind/"
< Server: occi-server/1.0 (linux) OCCI/1.0
< Connection: close
<
< <?xml version="1.0" encoding="UTF-8"?>
< <Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
<           xmlns:ovf="http://schemas.dmtf.org/ovf/envelope/1"
<           xmlns="http://schemas.dmtf.org/ovf/envelope/1"
<           xml:lang="en-US">
...
```

Next Steps

- Get involved in the working group at http://www.occi-wg.org/!
- Get the word out: blog about it, tweet about it (#occi), talk about it
- Join the mailing list and contribute to the discussion:
  - Seeking rough consensus & running code: ‘{}’ vs ‘<>’, above/below CRLF?
- Create a GridForge account & request access to the Google Code project
- Contact me if you have any questions, comments or concerns
  - Sam Johnston <samj@samj.net> or http://twitter.com/samj